Symantec AntiVirus™ Scan Engine Software Developer's Guide

Symantec AntiVirus™ Scan Engine Software Developer's Guide

Documentation version 4.1

Copyright Notice

Trademarks

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

SYMANTEC SOFTWARE LICENSE AGREEMENT
ENTERPRISE ANTIVIRUS SOFTWARE

THIS LICENSE AGREEMENT SUPERSEDES THE LICENSE AGREEMENT CONTAINED IN THE SOFTWARE INSTALLATION AND DOCUMENTATION.

SYMANTEC CORPORATION AND/OR ITS SUBSIDIARIES ("SYMANTEC") IS WILLING TO LICENSE THE SOFTWARE TO YOU AS AN INDIVIDUAL, THE COMPANY, OR THE LEGAL ENTITY THAT WILL BE UTILIZING THE SOFTWARE (REFERENCED BELOW AS "YOU OR YOUR") ONLY ON THE CONDITION THAT YOU ACCEPT ALL OF THE TERMS OF THIS LICENSE AGREEMENT. READ THE TERMS AND CONDITIONS OF THIS LICENSE AGREEMENT CAREFULLY BEFORE USING THE SOFTWARE. THIS IS A LEGAL AND ENFORCEABLE CONTRACT BETWEEN YOU AND THE LICENSOR. BY OPENING THIS PACKAGE, BREAKING THE SEAL, CLICKING ON THE "AGREE" OR "YES" BUTTON OR OTHERWISE INDICATING ASSENT ELECTRONICALLY, OR LOADING THE SOFTWARE, YOU AGREE TO THE TERMS AND CONDITIONS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO THESE TERMS AND CONDITIONS, CLICK ON THE "I DO NOT AGREE" OR "NO" BUTTON, OR OTHERWISE INDICATE REFUSAL AND MAKE NO FURTHER USE OF THE SOFTWARE.

1. LICENSE:

The software and documentation that accompanies this license (collectively the "Software") is the proprietary property of Symantec or its licensors and is protected by copyright law. While Symantec continues to own the Software, You will have certain rights to use the quantity of the Software for which You have paid the applicable license fees after Your acceptance of this license. This license governs any releases, revisions, or enhancements to the Software that the Licensor may furnish to You. Except as may be modified by an applicable Symantec license certificate, license coupon, or license key (each a "License Module") that accompanies, precedes, or follows this license, Your rights and obligations with respect to the use of licensed copies of this Software are as follows:

YOU MAY:

A. use the Software in the manner described in the Software documentation and in accordance with the License Module. If the Software is part of an offering containing multiple Software titles, the aggregate number of copies You may use may not exceed the aggregate number of licenses indicated in the License Module, as calculated by any combination of licensed Software titles in such offering. Your License Module shall constitute proof of Your right to make such copies. If no License Module accompanies, precedes, or follows this license, You may make one copy of the Software You are authorized to use on a single machine;

B. make one copy of the Software for archival purposes, or copy the Software onto the hard disk of Your computer and retain the original for archival purposes;

C. use the Software on a network or to protect a network such as at the gateway or on a mail server, provided that You have a license to the Software for each computer that can access the network;

D. after written consent from Symantec, transfer the Software on a permanent basis to another person or entity, provided that You retain no copies of the Software and the transferee agrees to the terms of this license; and

E. use the Software in accordance with any additional permitted uses set forth in Section 8 below.

YOU MAY NOT:

A. copy the printed documentation which accompanies the Software;

B. sublicense, rent or lease any portion of the Software; reverse engineer, decompile, disassemble, modify, translate, make any attempt to discover the source code of the Software, or create derivative works from the Software;

C. use a previous version or copy of the Software after You have received a disk replacement set or an upgraded version. Upon upgrading the Software, all copies of the prior version must be destroyed;

D. use a later version of the Software than is provided herewith unless You have purchased corresponding maintenance and/or upgrade insurance or have otherwise separately acquired the right to use such later version;

E. use, if You received the software distributed on media containing multiple Symantec products, any Symantec software on the media for which You have not received a permission in a License Module;

F. use the Software in any manner not authorized by this license; nor

G. use the Software in any manner that contradicts any additional restrictions set forth in Section 8 below.

2. CONTENT UPDATES:

Certain Symantec software products utilize content that is updated from time to time (antivirus products utilize updated virus definitions; content filtering products utilize updated URL lists; some firewall products utilize updated firewall rules; vulnerability assessment products utilize updated vulnerability data, etc.; collectively, these are referred to as "Content Updates"). You may obtain Content Updates for any period for which You have purchased upgrade insurance for the product, entered into a maintenance agreement that includes Content Updates, or otherwise separately acquired the right to obtain Content Updates. This license does not otherwise permit You to obtain and use Content Updates.

3. LIMITED WARRANTY:

Symantec warrants that the media on which the Software is distributed will be free from defects for a period of sixty (60) days from the date of delivery of the Software to You. Your sole remedy in the event of a breach of this warranty will be that Symantec will, at its option, replace any defective media returned to Symantec within the warranty period or refund the money You paid for the Software. Symantec does not warrant that the Software will meet Your requirements or that operation of the Software will be uninterrupted or that the Software will be error-free.

THE ABOVE WARRANTY IS EXCLUSIVE AND IN LIEU OF ALL OTHER WARRANTIES, WHETHER EXPRESS OR IMPLIED, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS. THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS. YOU MAY HAVE OTHER RIGHTS, WHICH VARY FROM STATE TO STATE AND COUNTRY TO COUNTRY.

4. DISCLAIMER OF DAMAGES:

SOME STATES AND COUNTRIES, INCLUDING MEMBER COUNTRIES OF THE EUROPEAN ECONOMIC AREA, DO NOT ALLOW THE LIMITATION OR EXCLUSION OF LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES SO THE BELOW LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW AND REGARDLESS OF WHETHER ANY REMEDY SET FORTH HEREIN FAILS OF ITS ESSENTIAL PURPOSE, IN NO EVENT WILL SYMANTEC BE LIABLE TO YOU FOR ANY SPECIAL, CONSEQUENTIAL, INDIRECT OR SIMILAR DAMAGES, INCLUDING ANY LOST PROFITS OR LOST DATA ARISING OUT

OF THE USE OR INABILITY TO USE THE SOFTWARE EVEN IF SYMANTEC HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

IN NO CASE SHALL SYMANTEC'S LIABILITY EXCEED THE PURCHASE PRICE FOR THE SOFTWARE. The disclaimers and limitations set forth above will apply regardless of whether You accept the Software.

5. U.S. GOVERNMENT RESTRICTED RIGHTS:

RESTRICTED RIGHTS LEGEND. All Symantec products and documentation are commercial in nature. The software and software documentation are "Commercial Items", as that term is defined in 48 C.F.R. section 2.101, consisting of "Commercial Computer Software" and "Commercial Computer Software Documentation", as such terms are defined in 48 C.F.R. section 252.227-7014(a)(5) and 48 C.F.R. section 252.227-7014(a)(1), and used in 48 C.F.R. section 12.212 and 48 C.F.R. section 227.7202, as applicable. Consistent with 48 C.F.R. section 12.212, 48 C.F.R. section 252.227-7015, 48 C.F.R. section 227.7202 through 227.7202-4, 48 C.F.R. section 52.227-14, and other relevant sections of the Code of Federal Regulations, as applicable, Symantec's computer software and computer software documentation are licensed to United States Government end users with only those rights as granted to all other end users, according to the terms and conditions contained in this license agreement. Manufacturer is Symantec Corporation, 20330 Stevens Creek Blvd., Cupertino, CA 95014, United States of America.

6. EXPORT REGULATION:

Export or re-export of this Software is governed by the laws and regulations of the United States and import laws and regulations of certain other countries. Export or re-export of Software to any entity on the Denied Parties List and other lists promulgated by various agencies of the United States Federal Government is strictly prohibited.

7. GENERAL:

If You are located in North America or Latin America, this Agreement will be governed by the laws of the State of California, United States of America. Otherwise, this Agreement will be governed by the laws of England. This Agreement and any related License Module is the entire agreement between You and Symantec relating to the Software and: (i) supersedes all prior or contemporaneous oral or written communications, proposals and representations with respect to its subject matter; and (ii) prevails over any conflicting or additional terms of any quote, order, acknowledgment or similar communications between the parties. This Agreement shall terminate upon Your breach of any term contained herein and You shall cease use of and destroy all copies of the Software. The disclaimers of warranties and damages and limitations on liability shall survive termination. The original of this Agreement has been written in English and English is the governing language of this Agreement. This Agreement may only be modified by a License Module which accompanies this license or by a written document which has been signed by both You and Symantec. Should You have any questions concerning this Agreement, or if You desire to contact Symantec for any reason, please write to: (i) Symantec Customer Service, 555 International Way, Springfield, OR 97477, U.S.A. or (ii) Symantec Customer Service Center, PO BOX 5689, Dublin 15, Ireland.

8. ADDITIONAL RESTRICTIONS FOR SPECIFIED SOFTWARE:

A. If the Software You have licensed is a specified Symantec AntiVirus™ for a third-party product or platform, You may only use that specified Software with the corresponding product or platform. You may not allow any computer to access the Software other than a computer using the specified product or platform. In the event that You wish to use the Software with a certain product or platform for which there is no specified Software, You may use the Symantec AntiVirus Scan Engine.

B. If the Software you have licensed is Symantec AntiVirus for NetApp® Filer, the following additional use(s) and restriction(s) apply:

i) You may use the Software only with a NetApp Filer server;

ii) You may use the Software only with files accessed through a NetApp Filer; and

iii) You may not use the Software on a server that exceeds the specified capacity set forth in Your License Module.

C. If the Software you have licensed is Symantec AntiVirus for Web Servers, the following additional use(s) and restriction(s) apply:

i) You may use the Software only with files that are received from third parties through a Web server;

ii) You may use the Software only with files received from less than 10,000 unique third parties per month; and

iii) You may not charge or assess a fee for use of the Software for Your internal business.

D. If the Software You have licensed is Symantec Web Security, independent of version or operating platform designation, upon the expiration of Your right to acquire Content Updates, the filtering definitions corresponding with all previous Content Updates will be entirely deleted and will no longer be available for use with the Software. Upon the expiration of Your right to acquire Content Updates, access to updated virus definitions will no longer be available. However, You may continue to use virus definitions previously acquired.

E. If the Software You have licensed is Symantec AntiVirus Corporate Edition, You may not use the Software on or with devices on Your network running embedded operating systems specifically supporting network-attached storage functionality without separately licensing a version of such Software specifically licensed for a specific type of network-attached storage device under a License Module.

F. If the Software You have licensed is Symantec AntiVirus for EMC® Celerra™ File Server, You may use the Software only with EMC Celerra servers and only if You have a license to the Software for each Celerra AntiVirus Agent (CAVA) associated with each such server. You may not allow any computer to access the Software other than an EMC Celerra server.

NetApp is a registered trademark of Network Appliance, Inc., in the U.S. and other countries.

EMC and Celerra are trademarks or registered trademarks of EMC Corporation in the U.S. and other countries.

Contents

Chapter 1

Getting started

This chapter includes the following topics:

About the Symantec AntiVirus Scan Engine

The Symantec AntiVirus Scan Engine is a network-accessible virus scanning and repair engine. The Symantec AntiVirus Scan Engine features all of the key virus-scanning technologies available in the complete line of Symantec antivirus products, making the Symantec AntiVirus Scan Engine one of the most effective virus solutions available for detecting and preventing virus attacks.

The Symantec AntiVirus Scan Engine provides virus scanning and repair capabilities for any application on an IP network, regardless of platform, using one of several supported protocols. Any application can pass files to the Symantec AntiVirus Scan Engine for scanning, which in turn scans the files for viruses and returns a cleaned file if necessary.

For more information about supported virus detection technologies and virus protection, see the *Symantec AntiVirus Scan Engine Implementation Guide*.

About the software developer's guide

Software developers can use the information provided in the *Symantec AntiVirus Scan Engine Software Developer's Guide* to create client applications that let third-party applications integrate with the Symantec AntiVirus Scan Engine for virus scanning and repair services. Client applications can communicate with the Symantec AntiVirus Scan Engine using one of several supported protocols. However, the only protocol that is supported in the software developer's guide is the Internet Content Adaptation Protocol (ICAP) version 1.0 presented in RFC 3507 (April 2003).

What's new in version 4.1.0 of the software developer's guide

The *Symantec AntiVirus Scan Engine Software Developer's Guide* includes the following new features:

- The underlying protocol in the Symantec AntiVirus Scan Engine application programming interface (API) has been changed to ICAP 1.0. Prior to this release, the underlying protocol in the Symantec AntiVirus Scan Engine API C library was the Symantec AntiVirus Scan Engine native protocol.

  See "About the change from the native protocol to ICAP" on page 12.

- The functionality that permits a client application to pass a full path rather than the actual file to the Symantec AntiVirus Scan Engine has changed temporarily for this release. The normal, optimized functionality will be restored in the next release.

  See "Maximizing performance" on page 14.

Integrating with the Symantec AntiVirus Scan Engine

Using ICAP as the communication protocol, software developers can create client applications (connectors) that let third-party applications integrate with the Symantec AntiVirus Scan Engine for virus scanning and repair services.

You can create a custom integration in one of two ways using ICAP:

- Use the Symantec AntiVirus Scan Engine application programming interface (API), which is provided as part of this software developer's kit.

  The underlying protocol for the scan engine API is ICAP 1.0.

  See "Constructing clients using the client API library" on page 49.

- Construct your own ICAP 1.0 client for the Symantec AntiVirus Scan Engine.

  See "Constructing clients using ICAP 1.0" on page 27.

About ICAP

ICAP is a lightweight protocol that was originally created for executing a remote procedure call on HTTP messages. ICAP is part of an evolving architecture that lets corporations, carriers, and Internet service providers (ISPs) dynamically scan, change, and augment data as it flows through ICAP servers. The protocol lets ICAP clients pass data to ICAP servers for adaptation (some sort of transformation or other processing, such as virus scanning). The server executes

its transformation service on the data and responds to the client, possibly with modified content.

In a typical integration for processing HTTP traffic, a caching proxy server retrieves the requested information from the Web. At the same time, it caches the information (stores a copy on disk) and, where possible, serves multiple requests for the same Web content from the cache. A caching proxy server can use ICAP to communicate with the Symantec AntiVirus Scan Engine and request that content that is retrieved from the Web be scanned and repaired.

About the API

You can use the client-side API C library to configure an application to pass files to the Symantec AntiVirus Scan Engine for scanning. The API includes 12 libraries: both static and dynamic libraries for each supported platform. The API library consists of 10 functions that provide scanning and repair services to client applications. The Symantec AntiVirus Scan Engine API C library is available for Red Hat Linux 7.2 and later (using either gcc 2.95 or 3.2), Solaris 7 and later (using either gcc 2.95 or 3.2), and Windows 2000 Server/Advanced Server (using either Microsoft Visual Studio 6 or 7). The provided C header file is included.

See "Configuring the Symantec AntiVirus Scan Engine for custom integrations" on page 17 and "Constructing clients using the client API library" on page 49.

About the change from the native protocol to ICAP

In version 4.1.0 of the software developer's guide, the underlying protocol in the Symantec AntiVirus Scan Engine application programming interface (API) has been changed to ICAP 1.0. Prior to this release, the underlying protocol in the Symantec AntiVirus Scan Engine API C library was the Symantec AntiVirus Scan Engine native protocol. To the degree possible, the change is transparent to the client application.

If you have an existing custom integration using the native protocol, the native protocol libraries can be found in the Scan_Engine_Integration_Kit/Old_API/ directory in the distribution package, but you should plan to change to ICAP 1.0.

About other protocols

The Symantec AntiVirus Scan Engine supports its own native protocol. This protocol is no longer supported in the software developer's kit. The Symantec AntiVirus Scan Engine also supports the ICAP 0.95 and Remote Procedure Call (RPC) 1.0 and 1.1 for NetApp Filer. These protocols are proprietary implementations and are not documented by Symantec.

About licensing

Key features for the Symantec AntiVirus Scan Engine, including antivirus scanning and virus definitions updates, are activated by license. After you install the Symantec AntiVirus Scan Engine, you install licenses through the Symantec AntiVirus Scan Engine administrative interface. When a license expires (for example, when a virus definitions product update subscription expires), a new license must be installed to renew the subscription. When no license is installed, functionality is limited. A license affects the relevant behavior only. For example, when no antivirus scanning license is installed, an administrator can access the administrative interface to view and modify settings and run reports, but no antivirus scanning is performed. When no virus definitions update license is installed, new virus definitions updates are not downloaded to keep protection current.

For more information about licensing, see the *Symantec AntiVirus Scan Engine Implementation Guide*.

Considerations for implementation

The Symantec AntiVirus Scan Engine can be easily implemented in an existing infrastructure. The Symantec AntiVirus Scan Engine runs on Solaris, Red Hat Linux, and Windows 2000 Server/Advanced Server platforms, and can be run on the same or a different computer than the client application.

You should consider a number of issues when you develop a client application for integration with the Symantec AntiVirus Scan Engine for virus scanning and repair services.

Deciding how to implement scanning

You can create a custom integration in one of two ways using ICAP:

- Use the Symantec AntiVirus Scan Engine application programming interface (API).

  See "Constructing clients using the client API library" on page 49.

- Construct your own ICAP client for the Symantec AntiVirus Scan Engine using ICAP 1.0.

  See "Constructing clients using ICAP 1.0" on page 27.

The Symantec AntiVirus Scan Engine API supports both file-based and stream-based scanning.

See "File-based scanning" on page 54.

See "Stream-based scanning" on page 54.

Maximizing performance

In a typical configuration, files are passed to the Symantec AntiVirus Scan Engine via a socket over the network because the scan engine is running on a separate computer. Depending on the network setup, client applications (applications that have been configured to pass files to the scan engine for scanning) can pass a full path rather than the actual file to the Symantec AntiVirus Scan Engine for improved performance.

Note: This functionality has changed temporarily for the 4.1.0 release of the API. Although client applications can pass a full path rather than the actual file to the Symantec AntiVirus Scan Engine, for this release the API passes all data that is to be scanned to the scan engine regardless of the configuration.

In the previous release, files to be scanned could be located on a drive that can be mounted over the network, such as a shared drive in Windows or a network file system (NFS) drive. If the client application and the scan engine had access to a shared directory, the client application could place the file in the shared directory and pass the full path to the Symantec AntiVirus Scan Engine, which could access the file directly. For cases in which the client application ran on the same computer as the Symantec AntiVirus Scan Engine, the client application could pass the file name to the scan engine, and the scan engine could open the file and scan it in place on the computer.

For the current release, the API continues to accept all parameters, and the exchange between the client application and the Symantec AntiVirus Scan Engine has not changed. However, even if you have specified only a full path, all data to be scanned is passed in its entirety to the scan engine for scanning. If you have an existing implementation, you do not need to change anything. If you are creating a custom integration, set up the client application to pass a full path to the scan engine. The normal, optimized functionality will be restored in a future release.

About automatic load balancing

The Symantec AntiVirus Scan Engine API provides scheduling across any number of computers that are running the Symantec AntiVirus Scan Engine. Client applications that pass files to the scan engine benefit from load-balanced

virus scanning without any additional effort. When multiple scan engines are used, the API determines the appropriate Symantec AntiVirus Scan Engine to receive the next file to be scanned, based on the scheduling algorithm.

If a Symantec AntiVirus Scan Engine is unreachable or stops responding during a scan, another scan engine is called and the faulty scan engine is taken out of rotation for a period of time (30 seconds is the default). If all of the scan engines are out of rotation, the faulty scan engines are called again. The API does not stop trying to contact the scan engines unless five engines do not respond or it appears that a file that is being scanned might have caused more than one engine to stop responding.

Where to start

Complete the following steps to configure client applications to use ICAP 1.0 to pass files to the scan engine for scanning:

- Become familiar with the design and features of the software.
  In addition to this guide, see the *Symantec AntiVirus Scan Engine Implementation Guide*.
- Decide how to deploy the Symantec AntiVirus Scan Engine on your network to meet your specific requirements.
  See "Considerations for implementation" on page 13.
- Install and configure the Symantec AntiVirus Scan Engine appropriately to use ICAP as the communication protocol.
  See the *Symantec AntiVirus Scan Engine Implementation Guide*.
- Install the API libraries.
  This guide provides the necessary information.
- Configure the client applications that will send files to the Symantec AntiVirus Scan Engine for scanning.

Chapter 2

Configuring the Symantec AntiVirus Scan Engine for custom integrations

This chapter includes the following topics:

Considerations for custom integration

The Symantec AntiVirus Scan Engine is designed to be easily integrated into any environment to provide antivirus scanning for any application. Client applications are configured to pass files to the Symantec AntiVirus Scan Engine, which scans the files for viruses and returns cleaned files if necessary.

The Symantec AntiVirus Scan Engine supports custom integration, in which a software developer creates a client application (connector) that provides virus scanning and repair services for a third-party application. The client application communicates with the Symantec AntiVirus Scan Engine using ICAP 1.0.

The Symantec AntiVirus Scan Engine must be configured to support the custom integration. This includes selecting ICAP as the communication protocol, configuring the ICAP-specific options, and selecting the types of files to scan.

You must decide how you plan to configure the client application and the scan engine to ensure that the appropriate files are scanned for viruses. This decision can depend on the capabilities of the third-party application. For example, the client application can decide what to scan and pass only the appropriate files to the scan engine. In other cases, the client application can be configured to pass all files to the scan engine, and the Symantec AntiVirus Scan Engine can be configured to scan those file types that are likely to contain viruses.

The client application must be configured to communicate with the Symantec AntiVirus Scan Engine and to handle the results that are returned from the scan engine. How the application is configured to handle the results that are returned from the scan engine can depend on the capabilities of the third-party application, including blocking access to infected files, quarantining unrepairable files, and so on.

See "Constructing clients using the client API library" on page 49 or "Constructing clients using ICAP 1.0" on page 27.

Configuring the scan engine to use ICAP

The Symantec AntiVirus Scan Engine must be configured to use ICAP to communicate with clients that are running either the proprietary version 0.95 of ICAP or version 1.0 (RFC 3507, April 2003). Any appropriate client can use ICAP to communicate with the Symantec AntiVirus Scan Engine to request the scanning and repair of files.

You can configure multiple client applications that use different versions of ICAP to pass files to a single Symantec AntiVirus Scan Engine. When you select ICAP as the communication protocol for the scan engine, the scan engine determines the

appropriate version of ICAP to use based on the header data passed in from the client application.

If you select ICAP as the protocol to be used by the Symantec AntiVirus Scan Engine, you must configure several ICAP-specific options. You must also configure the ICAP client to work with the Symantec AntiVirus Scan Engine. Table 2-1 details the information that must be provided.

Table 2-1 Protocol-specific options for ICAP

Option	Description
Scan Engine bind address	By default, the Symantec AntiVirus Scan Engine binds to all interfaces. You can restrict access to a specific interface by entering the appropriate bind address. You can use 127.0.0.1 (the loopback interface) to let only clients that are running on the same computer connect to the Symantec AntiVirus Scan Engine.
Port number	The port number must be exclusive to the Symantec AntiVirus Scan Engine. The default port number is 1344. If you change the port number, use a number that is not in use by any other program or service.
HTML message displayed for infected files	The Symantec AntiVirus Scan Engine includes a default HTML message to display to users when access to a file is denied because it contains a virus. You can customize this message by specifying an alternate path and file name or by editing the existing file. If you choose to edit the existing file, you do not have to change this setting.
ICAP scan policy	When an infected file is found, the Symantec AntiVirus Scan Engine can do any of the following: ■ Scan only: Deny access to infected files (but do nothing to the infected file). ■ Scan and delete: Delete all infected files, including files that are embedded in archive files, without attempting repair. ■ Scan and repair files: Attempt to repair infected files (but do nothing to files that cannot be repaired). Deny access to any files that cannot be repaired. ■ Scan and repair or delete: Attempt to repair infected files, and delete any unrepairable files from archive files. Deny access to any top-level files that cannot be repaired.

To configure the scan engine to use ICAP

1. On the Symantec AntiVirus Scan Engine administrative interface, in the left pane, click **Configuration**.
2. On the Protocol tab, click **ICAP**.
   The configuration settings display for the selected protocol.
3. In the Scan Engine bind address box, type a bind address, if necessary.
   By default, the Symantec AntiVirus Scan Engine binds to all interfaces. You can restrict access to a specific interface by typing the appropriate bind address. Use 127.0.0.1 (the loopback interface) to let only clients that are running on the same computer connect to the Symantec AntiVirus Scan Engine.
4. In the Port number box, type the TCP/IP port number to be used by client applications to pass files to the Symantec AntiVirus Scan Engine for scanning.
   The default setting for ICAP is port 1344.
5. In the HTML message displayed for infected files box, type the path and file name to supply an alternate HTML file, if necessary.
6. In the ICAP scan policy drop-down list, select how you want the Symantec AntiVirus Scan Engine to handle infected files.
   The default setting is Scan and repair or delete.
7. Click **Confirm Changes** to save the configuration.
8. Do one of the following:
   - Click **Continue** to make additional changes to the Symantec AntiVirus Scan Engine configuration.
   - Click **Restart** to save your changes and restart the Scan Engine service now.
   - Click **Save/No Restart** to save your changes. (Changes will not take effect until the service is restarted.)

Specifying file types to scan

You can control which file types are scanned by the Symantec AntiVirus Scan Engine by specifying extensions that you do not want to scan (using an exclusion list) or by specifying extensions that you want to scan (using an inclusion list), or you can scan all file types regardless of extension.

The Symantec AntiVirus Scan Engine is configured by default to scan all files except those with the extensions listed in a prepopulated exclusion list. The default exclusion list contains those file types that are unlikely to contain viruses, but you can edit this list.

Inclusion and exclusion lists by definition do not scan all file types; thus, new types of viruses may not always be detected. Scanning all files regardless of extension is the most secure setting, but imposes the heaviest demand on resources.

Note: During virus outbreaks, you may want to scan all files even if you normally control the file types that are scanned with the inclusion or exclusion list.

Using an inclusion list to control which types of files are scanned is the least secure setting. Only those files types that are specifically listed in an inclusion list are scanned; thus, with an inclusion list, there is an almost limitless number of possible file extensions that are not scanned. For this reason, the inclusion list is not prepopulated, but you can choose to populate this list if you want to limit the file types that are scanned using an inclusion list.

If you use either the inclusion or the exclusion list to control the file types that are scanned (rather than scanning all files), the manner in which the list is applied differs depending on which of the following protocols are in use by the Symantec AntiVirus Scan Engine (these differences are particularly important if you are upgrading from ICAP 0.95 to ICAP 1.0):

- Native protocol and ICAP 1.0: The inclusion or exclusion list is used by the Symantec AntiVirus Scan Engine only to determine which files to scan of those that are embedded in archival file formats (for example, .zip or .lzh files). All top-level files that are sent to the Symantec AntiVirus Scan Engine are scanned regardless of file extension.

  Note: If you are using the native protocol or ICAP 1.0 and want to control the file types that are scanned at the top level, you must provide logic or take advantage of existing mechanisms on the client side to send only certain file types to the Symantec AntiVirus Scan Engine for scanning. The logic on the client side controls the types of files that are scanned at the top level, and the extension list setting controls which embedded files are scanned.

- ICAP 0.95: The inclusion or exclusion list applies to all files that are sent to the Symantec AntiVirus Scan Engine for scanning. This extension list is consulted for both top-level files and embedded files that are contained in archival file formats (for example, .zip or .lzh files).

Specify which file types to scan

You can control which file types are scanned by specifying extensions that you want to scan or that you do not want to scan, or you can scan all files regardless of extension. The Symantec AntiVirus Scan Engine is configured by default to scan all files except those with extensions listed in the prepopulated exclusion list.

To scan all files regardless of extension

1. On the Symantec AntiVirus Scan Engine administrative interface, in the left pane, click **Blocking Policy**.
2. On the AntiVirus tab, under File types to be scanned, click **Scan all files regardless of extension**.
3. Click **Confirm Changes** to save the configuration.
4. Do one of the following:
   - Click **Continue** to make additional changes to the Symantec AntiVirus Scan Engine configuration.
   - Click **Restart** to save your changes and restart the Scan Engine service now.
   - Click **Save/No Restart** to save your changes. (Changes will not take effect until the service is restarted.)

To scan all files except for those with extensions that are in the exclusion list

1. On the Symantec AntiVirus Scan Engine administrative interface, in the left pane, click **Blocking Policy**.
2. On the AntiVirus tab, under File types to be scanned, click **Scan all files except those with the following extensions**.
3. Edit the extension list to add extensions that you do not want to scan or delete extensions that you want to scan.

   Use a period with each extension in the list. Separate each extension with a semicolon (for example, .com;.doc;.bat). To exclude files with no extension, use two adjacent semicolons (for example, .com;.exe;;).
4. To restore the default extension lists, click **Restore default lists**.

5 Click **Confirm Changes** to save the configuration.

6 Do one of the following:

- Click **Continue** to make additional changes to the Symantec AntiVirus Scan Engine configuration.
- Click **Restart** to save your changes and restart the Scan Engine service now.
- Click **Save/No Restart** to save your changes. (Changes will not take effect until the service is restarted.)

To scan only files with extensions that are in the inclusion list

1 On the Symantec AntiVirus Scan Engine administrative interface, in the left pane, click **Blocking Policy**.

2 On the AntiVirus tab, under File types to be scanned, check **Scan files with the following extensions**.

3 Edit the extension list to add extensions that you want to scan or delete extensions that you do not want to scan.

Use a period with each extension in the list. Separate each extension with a semicolon (for example, .com;.doc;.bat). To scan files that have no extensions, use two adjacent semicolons (for example, .com;.exe;;).

4 Click **Confirm Changes** to save the configuration.

5 Do one of the following:

- Click **Continue** to make additional changes to the Symantec AntiVirus Scan Engine configuration.
- Click **Restart** to save your changes and restart the Scan Engine service now.
- Click **Save/No Restart** to save your changes. (Changes will not take effect until the service is restarted.)

Changing an ICAP response

In its default configuration when ICAP is in use as the communication protocol, the Symantec AntiVirus Scan Engine sends a 200 OK response for the following scenarios:

- No virus was detected during the scan.
- A virus was detected, but the file could not be repaired.
- A virus was detected and the Symantec AntiVirus Scan Engine is configured for scan only mode.

If a virus is detected and the file cannot be repaired or the scan engine is configured for scan only mode, the Symantec AntiVirus Scan Engine includes a replacement file (the file specified for ICAPInfectionHTMLfile) in the body of the response message. The replacement file contains an HTML and/or email message that informs users that their access to a file is being denied because it contains a virus.

The Symantec AntiVirus Scan Engine default behavior deviates from the ICAP 1.0 standard, which does not support the automatic sending of a replacement file. In the ICAP 1.0 standard, this type of context-sensitive behavior is performed by the client rather than the scan engine.

The ICAP 1.0 standard also differs from the ICAP 0.95 implementation in this regard. If you were using ICAP 0.95 and want to maintain similar functionality when you migrate to ICAP 1.0, you should use the Symantec AntiVirus Scan Engine default ICAP response setting if the client application can support it.

If your client application closely follows the ICAP 1.0 standard, you might need to change the Symantec AntiVirus Scan Engine default ICAP response setting to receive an ICAP 403 response instead of a replacement file. The scan engine sends a 403 response if the file is infected and cannot be repaired. If no virus is detected, the scan engine returns a 200 OK response. You should change the default ICAP response setting only if you are sure that the client application supports this behavior.

Note: To make this change, you must edit the Symantec AntiVirus Scan Engine configuration file.

To change the ICAP response

1. Locate the Symantec AntiVirus Scan Engine configuration file.

   The configuration file is located in the following default directories:

   - On Solaris and Red Hat Linux: /opt/SYMCScan/etc/symcscan.cfg
   - On Windows 2000 Server/Advanced Server: C:\Program Files\Symantec\Scan Engine\symcscan.cfg

   If you are running more than one copy of the Symantec AntiVirus Scan Engine on a computer, make sure that you have the appropriate configuration file.

2. Open the configuration file with a text editor.
3. At ICAPResponse=, type one of the following to specify the scan engine response when a file is blocked because it is unrepairable (ICAP 1.0 only):

   - 0: Send an ICAP 403 response.
   - 1: Send a replacement file.

     This is the default setting.

4. Save the file.
5. Stop and restart the Symantec AntiVirus Scan Engine.

Chapter 3

Constructing clients using ICAP 1.0

This chapter includes the following topics:

How ICAP works

The Internet Content Adaptation Protocol (ICAP) is a request/response-based protocol that lets ICAP clients pass messages to ICAP servers for processing or adaptation. The client initiates the session by sending request messages over a TCP/IP connection to a passively waiting ICAP server on a designated port. (Port 1344 is the default ICAP port.) The server then runs the service that was requested, such as antivirus scanning; performs any transformations that are necessary, such as repairing an infected file; and sends a response back to the client with any modified data.

A single transport can be used for multiple request/response pairs. Requests are matched with responses by allowing only one outstanding request on a connection at a time. Multiple connections can be used.

The Symantec AntiVirus Scan Engine supports the ICAP response modification mode (RESPMOD). In response modification mode, an ICAP client receives a data request from an origin server. The ICAP client then passes the data to an ICAP server for evaluation and post-processing.

The Symantec AntiVirus Scan Engine supports the following ICAP methods (also called commands):

- OPTIONS: Lets the client obtain information from an ICAP server about available services
  See "Determining which services are supported" on page 33.
- RESPMOD: Lets the client send files to the Symantec AntiVirus Scan Engine for scanning and repair services
  See "Sending files for scanning" on page 36.

Finding more information on ICAP

The Symantec AntiVirus Scan Engine supports the ICAP 1.0 specification presented in RFC 3507 (April 2003). Much of the information in the *Symantec AntiVirus Scan Engine Software Developer's Guide* is obtained directly from the specification. However, the specification contains more extensive examples and additional detailed information. Developers are encouraged to consult the specification, as well as other sources.

For more information about ICAP specifications, visit the ICAP Web site at: www.i-cap.org

About ICAP messages

ICAP clients and servers communicate through messages, which are similar in format to HTTP. ICAP messages consist of client requests and server responses. All ICAP messages consist of a start line, which includes a client request or server response (depending on the type of message), header fields, and the message body. A blank line must precede the message body to distinguish the headers from the message body.

Multiple messages can be encapsulated in a single ICAP message for vectoring of requests, responses, and request/response pairs on an ICAP server. Encapsulated messages must include an encapsulated header, which offsets the start of each encapsulated section from the start of the message body.

See "About encapsulated messages" on page 31.

Although request and response messages have unique headers, some headers are common to both requests and responses. Table 3-1 lists the general request/response headers that the Symantec AntiVirus Scan Engine uses.

Table 3-1 General request/response headers

Header	Description
Connection	Specifies options that the message sender wants to use only for that connection and not for proxies over other connections. For example: Connection: close
Date	Provides the date and time that the message was created, using standard HTTP date and time format. For example: Date: Tue, 5 Nov 2002 14:29:31 GMT

About request messages

All ICAP client requests must start with a request line that includes the following components:

- Method: ICAP command or operation to perform (for example, RESPMOD)
- Uniform Resource Identifier (URI): Complete host name of the ICAP server and the path of the resource that is being requested
- ICAP version: Version string for the current version of ICAP using the format ICAP/version number (for example ICAP/1.0)

The URI consists of the following components:

```
ICAP_URI = Scheme ":" Net_Path [ "?" Query ]

Scheme = "icap"

Net_Path = "//" Authority [ Abs_Path ]

Authority = [ userinfo "@" ] host [ ":" port ]
```

Header fields follow the request line and specify the ICAP resource that is being requested as well as other information, such as cache control information. ICAP-specific headers are encapsulated. The header fields end with a blank line followed by the message body. The message body contains the encapsulated HTTP messages that are being sent for scanning and modification.

Table 3-2 lists the specific request headers that are allowed in ICAP requests.

Table 3-2 Request headers

Header	Description
Allow	Lists the methods that the resource supports. For example, a client request can include an Allow: 204 header, which indicates that it will allow the server to reply to the message with a 204 No Content response if the file does not need modification. The client must buffer the message.
From	Provides the Internet email address for the user who is sending the client request. The address should use the standard HTTP mailbox format. For example: From: username@symantec.com
Host	Specifies the host name and port number of the resource being requested.
Referer	Specifies the path that the client followed to obtain the URI. This optional header lets the server generate lists of back-links to resources and trace invalid links.
User-Agent	Identifies the software program that is used by the client that originated the request. This information is used for statistical purposes, to trace protocol violations, and to tailor responses to the software capabilities.
Preview	(ICAP-specific header) Lets the client send a portion of a file to the Symantec AntiVirus Scan Engine for scanning. The client uses this header to specify the amount of data, in bytes, that will be sent for preview.

About response messages

ICAP client responses start with a status line, which includes the ICAP version and a status code. For example:

```
ICAP/1.0 200 OK
```

Status codes vary depending on the type of request.

See Table 3-3 "OPTIONS response codes" on page 34 and Table 3-6 "RESPMOD response codes for successfully processed scanning" on page 39 for more information about status codes.

The status line is followed by one or more response headers that let the server pass additional information (that is, information that cannot be placed in the status line) to the client.

See "About RESPMOD response headers" on page 41.

About encapsulated messages

The ICAP encapsulation model provides a lightweight means of packaging multiple HTTP message sections into a single ICAP message for vectoring of requests, responses, and request/response pairs on an ICAP server. An encapsulated section can consist of either HTTP message headers or bodies.

Encapsulated message bodies must be transferred using chunked transfer encoding. This keeps the transport-layer connection between the client and server open for later use and lets the server send incremental responses to reduce the latency that is perceived by users. Encapsulated headers are not chunked. This lets the ICAP client copy the header directly from the HTTP client to the ICAP server without having to reprocess it.

Note: The chunked transfer encoding modifies the body of a message so that it can be transferred as a series of chunks, each with its own (hexadecimal) size indicator, followed by an optional footer that contains entity-header fields. See the HTTP/1.1 specification (RFC 2616, section 3.6.1) for more information.

The encapsulated header must be included in every ICAP message, except for OPTIONS requests. This header provides information about where each encapsulated section and message body starts and ends.

For example:

```
Encapsulated: req-hdr=0, res-hdr=45, res-body=100
```

This example indicates that the message encapsulates a group of request headers, response headers, and a response body at 0, 45, and 100 byte offsets. Byte offsets

use a decimal format; however, chunk sizes within an encapsulated body use a hexadecimal format. If none of the message body is encapsulated, a null-body header is used.

Encapsulated headers use the following syntax:

```
encapsulated_header: "Encapsulated: " encapsulated_list

encapsulated_list: encapsulated_entity |

            encapsulated_entity ", " encapsulated_list

encapsulated_entity: reqhdr | reshdr | reqbody | resbody | optbody

reqhdr  = "req-hdr" "=" (decimal integer)

reshdr  = "res-hdr" "=" (decimal integer)

reqbody = { "req-body" | "null-body" } "=" (decimal integer)

resbody = { "res-body" | "null-body" } "=" (decimal integer)

optbody = { "opt-body" | "null-body" } "=" (decimal integer)
```

Encapsulated headers must end with a blank line to make them readable and to terminate line-by-line HTTP parsers.

About the antivirus scanning process

The Symantec AntiVirus Scan Engine provides basic antivirus scanning services to ICAP clients through the RESPMOD method.

The general process is as follows:

- A client application receives a request to access data.
- The client application forwards the data (or a portion of it for preview) to the Symantec AntiVirus Scan Engine for scanning using the RESPMOD method.
  See "Sending files for scanning" on page 36.
- The Symantec AntiVirus Scan Engine processes the request, performs any transformations that are necessary (such as repairing infected files), and sends a response back to the client with any modified data.

Before sending files, the client can query the ICAP server using the OPTIONS method to determine which services are supported.

See "Determining which services are supported" on page 33.

Determining which services are supported

The OPTIONS method lets a client application query an ICAP server for information about supported services and commands and preferred file handling methods. The client application should perform this query before sending files for scanning.

The OPTIONS method consists of a request line that contains the address or name of the server on which the Symantec AntiVirus Scan Engine is installed and the URI for the Symantec AntiVirus Scan Engine service that you want to query.

When the Symantec AntiVirus Scan Engine receives an OPTIONS request from a client application, it sends a response that includes the following information:

- Maximum number of simultaneous connections allowed
- Preferred data preview size
- Preferred file handling methods
- Supported methods (for example, RESPMOD)

Querying the AVSCAN service

The AVSCAN service performs scanning, repair, and delete functions using the scanning preferences that you specify through the Symantec AntiVirus Scan Engine administrative user interface.

Use the following format to specify the URI:

icap://<Server>/avscan

A sample OPTIONS request is as follows:

```
OPTIONS icap://saves.com/avscan ICAP/1.0
Host: icapclient.savese.com
```

The AVSCAN service returns a response, which includes a status line followed by a series of response headers. A sample OPTIONS response is as follows:

```
ICAP/1.0 200 OK
Date: Fri Apr  4 03:06:25 2003 GMT
Methods: RESPMOD
Service: Symantec AntiVirus Scan Engine/4.0.3.37
ISTag: "1049425529"
X-Definition-Info: 20030106.006
Max-Connections: 16
X-Allow-Out: X-OuterContainer-Is-Mime, X-Infection-Found, X-
Violations-Found, X-Definition-Info, X-AV-License
X-Allow-Out: X-SAVSE-AV-Status
Allow: 204
```

```
Options-TTL: 3600
Preview: 4
Transfer-Preview: *
X-AV-License: 1
Encapsulated: null-body=0
```

This response informs the client that, besides the OPTIONS method, the only supported method is RESPMOD, that the optional 204 shortcut is supported, that four bytes of preview information are preferred, that all files should be sent for preview, and that the data in this response may be cached up to one hour. The maximum number of simultaneous connections that the Symantec AntiVirus Scan Engine supports may vary, depending on the operating environment. In this example, the server supports a maximum of 16 simultaneous connections.

See "OPTIONS response codes" on page 34 and "OPTIONS response headers" on page 35.

OPTIONS response codes

Table 3-3 lists the possible response codes to an OPTIONS request.

Table 3-3 OPTIONS response codes

Response code	Text	Description
200	OK	Server processed request successfully
400	Bad request	Syntax error or other problem parsing the request
404	Not found	URI in request does not correspond to an available service
405	Method not implemented	Not valid unless OPTIONS is misspelled
408	Request timeout	Client took too long to send request
500	Internal server error	Generic problem with server
503	Service unavailable/ overloaded	Server not ready to provide service
505	ICAP version not supported	Only ICAP 1.0 is supported with this method
551	Resource unavailable	Memory or disk problem on server

OPTIONS response headers

Table 3-4 lists the response headers that are included in an OPTIONS response.

Table 3-4 OPTIONS response headers

Header	Description
Date	Specifies the date and time, as set on the server clock.
Service	Specifies the name and version number of the ICAP server.
ISTag	(ICAP service tag) Lets an ICAP server send service-specific information to an ICAP client. This data can be used to validate whether a server response, including cached data, is still valid. The Symantec AntiVirus Scan Engine returns an ISTag with every response to indicate the time of the most recent change of virus definitions or scan policy. Cached data that does not match the current ISTag is no longer valid. The value is an integer that represents the number of seconds since the UNIX epoch. For example: ISTag: “99293223881”
Methods	Specifies the methods (commands) that are supported by the service that you queried.
Allow	Lists the optional ICAP features that the server supports.
Preview	Indicates the number of bytes of data that can be sent to the Symantec AntiVirus Scan Engine for preview.
Transfer-Preview	Lists the file extensions that should be sent to the Symantec AntiVirus Scan Engine for preview before sending the entire file. An asterisk (*) wildcard character represents the default behavior for all file extensions that are not specified in another Transfer type header.
Transfer-Complete	Lists the file extensions that should always be sent in their entirety to the Symantec AntiVirus Scan Engine and that should not be previewed. An asterisk (*) wildcard character represents the default behavior for all file extensions that are not specified in another Transfer type header.

Table 3-4 OPTIONS response headers

Header	Description
Max-Connections	Indicates the maximum number of simultaneous ICAP connections that the server supports.
Options-TTL	Indicates the time (in seconds) during which the response is valid or cached. A blank header indicates that the response does not expire.
Encapsulated	Offsets the start of each encapsulated section from the start of the message body.
X-AV-License	Indicates whether a valid AV scanning license has been installed on the scan engine, where 1 indicates a valid license and 0 indicates no valid license.
X-Allow-Out	Indicates the custom X-headers returned in responses from this scan engine.
X-Definition-Info	Indicates the date and revision number of the virus definitions in the following format: YYYYMMDD.RRR, where YYYY is the four-digit year, MM is the month, DD is the day, and RRR is the revision number.

Sending files for scanning

The Symantec AntiVirus Scan Engine supports one URI for scanning and repair services, which uses the following format:

icap://server.name:port/avscan

Replace server.name with the name of the server on which the Symantec AntiVirus Scan Engine is running. The port number is optional if the Symantec AntiVirus Scan Engine is running on port 1344, which is the default ICAP port.

Sending portions of files for preview

The Symantec AntiVirus Scan Engine can preview data to determine whether it needs to be scanned based on known virus behavior. For example, .gif files are typically not scanned because they generally do not contain executable code. The rules for which types of files are suitable for preview are determined by the inclusion and exclusion lists that are configured in the Symantec AntiVirus Scan Engine.

Before a file is sent for scanning, the client should send an OPTIONS request to determine whether a file type is suitable for preview and how much data should be sent. The Symantec AntiVirus Scan Engine provides this information in the following headers of the OPTIONS response message:

- Preview: Indicates the preferred number of bytes of data that can be sent
- Transfer-Complete: Indicates which file types should be sent in their entirety
- Transfer-Preview: Indicates which file types should be sent for preview

For more information about OPTIONS response headers, see Table 3-4 "OPTIONS response headers" on page 35.

Table 3-5 details the Symantec AntiVirus Scan Engine scanning behavior that is based on the configured scanning policies.

For more information, see the *Symantec AntiVirus Scan Engine Implementation Guide*.

Table 3-5 Scanning behavior

Scanning policy	Transfer-Complete header	Transfer-Preview header	Scanning behavior
Scan all files regardless of extension	Asterisk (*) character	Not used	The Symantec AntiVirus Scan Engine scans every file in its entirety without previewing it first.
Scan files with the following extensions (inclusion list)	List of file extensions	Asterisk (*) character	The Symantec AntiVirus Scan Engine scans the entire file for each file type that is in the inclusion list. Other file types are previewed for dangerous content.
Scan all files except those with the following extensions (exclusion list)	Asterisk (*) character	List of file extensions	The Symantec AntiVirus Scan Engine previews the file types that are listed in the Transfer-Preview header for dangerous content. All other file types, including unidentified file types, are scanned in their entirety.

If an OPTIONS response indicates that a file is suitable for preview, the client should include a Preview header in the request message that indicates the portion of data, in bytes, that is being sent for preview. The Symantec AntiVirus Scan

Engine then evaluates the initial chunk of data to determine whether a full scan is required. If so, the Symantec AntiVirus Scan Engine requests the remainder of the data. Scan results are returned in the RESPMOD response message.

See "Interpreting RESPMOD response messages" on page 38.

Allowing no content responses

The Allow: 204 header is an optional request header that lets the Symantec AntiVirus Scan Engine return a 204 No Content response code if the message does not require modification. This can optimize server performance because the Symantec AntiVirus Scan Engine can declare a file clean without waiting to receive the entire message and does not have to return the message. The processing burden is placed on the client, which must buffer the entire message during the scan. The Symantec AntiVirus Scan Engine returns a 204 No Content response outside of a preview only if the client request includes an Allow: 204 header.

Interpreting RESPMOD response messages

After performing a scan, the Symantec AntiVirus Scan Engine returns a response message, which indicates the scan results and, if applicable, any modified data. Response messages start with an ICAP status line, which includes the ICAP version and a status code. For example:

```
ICAP/1.0 200 OK
```

The status line is followed by one or more response headers that let the server pass additional information to the client that cannot be placed in the status line. If no virus is detected, the Symantec AntiVirus Scan Engine also returns a copy of the scanned data to the client, unless the client request includes an Allow: 204 header. If so, the Symantec AntiVirus Scan Engine only returns a response message. If a virus is detected, the scan policy settings determine whether the Symantec AntiVirus Scan Engine attempts to repair the file and whether modified content is returned to the client.

A standard RESPMOD request is as follows:

```
RESPMOD icap://saves.com/avscan ICAP/1.0
Host: icapclient.savese.com
Allow: 204
Preview: 4
Encapsulated: req-hdr=x res-hdr=y res-body=z

[ Request headers…]

[ Response headers…]

4

[ First four bytes of body data… ]

0
```

RESPMOD response codes

Response codes in the 200 class indicate whether a virus was found and which action, if any, was taken to repair the file. Response codes in the 400 and 500 classes are error codes. Table 3-6 lists the possible RESPMOD response codes for scans that processed successfully.

Table 3-6 RESPMOD response codes for successfully processed scanning

Response code	Text	Description
100	Continue	The Symantec AntiVirus Scan Engine completed a preview and requires additional data.
200	OK	The request was processed successfully. In its default configuration, the Symantec AntiVirus Scan Engine returns a 200 response if no virus was detected or if a virus was detected but either the file cannot be repaired or the scan engine is configured for scan only mode. If the Symantec AntiVirus Scan Engine is configured to send an ICAP 403 response, the Symantec AntiVirus Scan Engine returns a 200 response only if no virus was detected. See "Changing an ICAP response" on page 24.
201	Created	A virus was detected and the file has been repaired (not valid for scan only mode). The response also includes the repaired data.
204	No content necessary	No virus was detected and the client sent an Allow: 204 header, which indicates that the Symantec AntiVirus Scan Engine does not need to return data to the client.

Table 3-7 lists the possible RESPMOD response codes for client errors.

Table 3-7 Client error RESPMOD response codes

Response code	Text	Description
400	Bad request	The Symantec AntiVirus Scan Engine was unable to process the request because of a syntax error or other general problem in the client request.
403	Forbidden. Infected and not repaired.	The data was infected and cannot be repaired or the Symantec AntiVirus Scan Engine is configured for scan only mode. **Note:** This response is the standard ICAP behavior, as documented in the ICAP 1.0 specification. However, to support NetApp NetCache clients, the Symantec AntiVirus Scan Engine does not follow this behavior by default. If your client application closely follows the ICAP 1.0 standard, you might need to change the default setting for an ICAP response. See "Changing an ICAP response" on page 24.
404	Not found	The URI that was specified in the client request does not match any service that is available on the server.
405	Method not implemented	The client requested a method that is not supported by the server (other than OPTIONS or RESPMOD). The response also includes an Allow header line that identifies the supported methods.
408	Request timeout	The client took too long to send the request.

Table 3-8 lists the possible RESPMOD response codes for server errors.

Table 3-8 Server error RESPMOD response codes

Response code	Text	Description
500	Internal server error	A generic problem occurred on the server.
503	Service unavailable/ overloaded	The server is not ready to provide service.

Table 3-8 Server error RESPMOD response codes

Response code	Text	Description
505	ICAP version not supported	The ICAP client is using a version of ICAP other than 1.0.
533	Error scanning file	An error occurred during file scanning that prevented the Symantec AntiVirus Scan Engine from completing the scan.
539	Aborted - No AV scanning license	The Symantec AntiVirus Scan Engine is unable to scan the data because a valid license does not exist. See "About licensing" on page 13.
551	Resource unavailable	A memory or disk problem occurred on the server.

About RESPMOD response headers

Table 3-9 lists the RESPMOD response headers that are specific to antivirus scanning.

Table 3-9 RESPMOD response headers

Header	Description
Service	Provides information about the software that is used by the origin server to handle the request. If the request is handled by a proxy server, the proxy server can add information using the Via field, but cannot modify the server response.
Service-ID	Provides information about the software that is used by the origin server to handle the request. This header is a shorter version of the Service header.

Table 3-9 RESPMOD response headers

Header	Description
ISTag	(ICAP service tag) Lets an ICAP server send service-specific information to an ICAP client. This data can be used to validate whether a server response, including cached data, is still valid. The Symantec AntiVirus Scan Engine returns an ISTag with every response to indicate the time of the most recent change of virus definitions or scan policy. Cached data that does not match the current ISTag is no longer valid. The value is an integer that represents the number of seconds since the UNIX epoch. For example: ISTag: “99293223881”
X-Outer-Container-Is-Mime	(Optional header) An integer value that indicates whether the outer container is a valid MIME container. A client application can use this information to reject content that does not meet this criteria. Zero (0) indicates that the outer container is not a valid MIME container, and one (1) indicates that the outer container is a valid MIME container.
X-Infection-Found	Provides information about an infection that is found. Only one violation is reported, regardless of the number of violations found. The header provides the following information regarding the infection: ■ Violation type: An integer value for the violation. Zero (0) indicates a virus, one (1) indicates a mail policy violation, and two (2) indicates a container violation or malformity. ■ Resolution: An integer value that indicates what action was taken on the file. Zero (0) indicates that the file was not fixed, one (1) indicates that the file was repaired, and two (2) indicates that access to the file was blocked. ■ Threat value: String that describes the virus or violation that was found.

Table 3-9 RESPMOD response headers

Header	Description
X-Violations-Found	Indicates the total number of violations (either an infection or a policy violation) that were found in the scanned data. If violations were detected, the header is followed by a series of indented lines that provide the following information for each violation: ■ File name: The name of the scanned file or the name of a nested component within the scanned file, with each component name separated by a slash mark (/). ■ Violation name: The English-readable name of the violation. ■ Violation ID: A numeric code for the violation. ■ Disposition: An integer value that indicates what action was taken to fix the file. Zero (0) indicates that the file was not fixed, one (1) indicates that the file was repaired, and two (2) indicates that the file was deleted.

Sample responses

In the following examples, C indicates the client and S indicates the Symantec AntiVirus Scan Engine. Text throughout each section of the exchange explains what is happening.

Scanning a clean file with a 4-byte preview

The following example shows a request to scan a clean file with a 4-byte preview.

```
C: RESPMOD icap://192.168.1.2:1344/AVSCAN ICAP/1.0
C: Host: 192.168.1.2:1344
C: Preview: 4
C: Allow: 204
C: Encapsulated: req-hdr=0, res-hdr=84, res-body=131
C:
```

The preceding section of code is the ICAP header, which is sent from the client to the ICAP server (in this case, the Symantec AntiVirus Scan Engine). The RESPMOD method is used to request an antivirus scan using the default scanning mode that is configured on the scan engine.

The client uses preview mode and sends the first 4 bytes of data as the initial preview. The preview lets the scan engine determine the data type of the content to be scanned. Thus, files that may have had the file extension altered are still scanned for viruses if the data type is one that can potentially contain viruses.

The Allow: 204 header indicates that the client does not require a copy of clean data to be sent back from the antivirus scanner after the scan is complete. If modifications are made (for example, the file is repaired), the data is always returned.

The last line indicates that the headers from the original outgoing request are present and can be found at byte offset 0 in the body of this ICAP request, that the response headers from the content provider are included and start at byte offset 84 in the body of this ICAP request, and that the response body from the content provided is included and can be found starting at byte offset 131 of this ICAP request.

```
C: GET http://icapone.symantec.com/CLEAN01.DOC HTTP/1.1
C: Host: icapone.symantec.com
C:
```

The preceding section of code is the encapsulated copy of the original request that initiated the transaction. The file name near the end of the request line is used to identify the document for logging purposes on the scan engine.

```
C: HTTP/1.1 200 OK
C: Transfer-Encoding: chunked
C:
```

The preceding section of code shows response headers from the content provider. The scan engine does not use these headers, but the headers must accompany (possibly modified) any response that is sent back to the ICAP client, for eventual delivery to the user who initiated the request.

```
C: 4
[ 4-byte chunk ]
C:
C: 0
C:
```

In the preceding section of code, the ICAP client sends the 4-byte preview data chunk to the scan engine, as indicated in the ICAP headers. The scan engine examines the data and determines whether the remainder of the file needs to be scanned.

```
S: ICAP/1.0 100 Continue
S:
```

In the preceding section of code, the scan engine determines, based on the current policy, that the remainder of the data must be scanned, and sends a Continue message.

```
C: 790a
[ Chunk of size 30986 bytes ]
C:
C: 0
C:
```

In the preceding section of code, the remainder of the data is sent from the client to the scan engine. The chunked data encoding scheme is used. When the data is received by the scan engine, the scan is completed, and no infection is found.

```
S: ICAP/1.0 204 No Content Necessary
S: ISTag: "1049478470"
S: Date: Fri Apr 04 17:58:33 2003 GMT
S: Service: Symantec AntiVirus Scan Engine/4.0.4.41
S: Service-ID: SYMCScan/4.0.4.41
S: X-Outer-Container-Is-Mime: 0
```

In the preceding section of code, because the client indicated (if the content was clean) that the scan engine did not need to return a copy of the data, the 204 No Content Necessary result is returned to the client to indicate that no problem was found.

The ISTag header is returned so that the client can determine if the virus definitions have changed on the scan engine. This type of change might cause an ICAP client that implements caching to discard data or to mark data that needs to be scanned again for infection based on the new virus definitions.

The X-Outer-Container-is-Mime header is set to 0 to indicate that the top-level file scanned by the scan engine was not in MIME (email) format. Mail clients expect the top level to be in MIME format, and can use this feature to identify malformed messages that potentially should be discarded.

Scanning an infected file that cannot be repaired

The following example shows a request to scan a file that is infected and that cannot be repaired. This example also includes a 4-byte preview.

```
C: RESPMOD icap://192.168.1.2:1344/AVSCAN ICAP/1.0
C: Host: 192.168.1.2:1344
C: Preview: 4
C: Allow: 204
C: Encapsulated: req-hdr=0, res-hdr=84, res-body=131
C:
C: GET http://icapone.symantec.com/BHM97UR.DOC HTTP/1.1
C: Host: icapone.symantec.com
C:
C: HTTP/1.1 200 OK
C: Transfer-Encoding: chunked
C:
C: 4
[ 4 byte chunk ]
C:
C: 0
C:
S: ICAP/1.0 100 Continue
S:
C: 790a
```

In the preceding section of code, the remainder of the file is sent as a chunk of size 30986 bytes, with a 0 chunk at the end to signal the end of the file. After it has received all of the data, the scan engine scans the data, finds an infection, and reports the results to the client.

With the default settings, an ICAP 200 response is returned with new content in the body of the message to replace the original infected content. This is not in strict compliance with the ICAP specification, but works better with a number of WWW proxy servers.

For strict compliance, change the value of the ICAPResponse setting in the scan engine configuration file to 0 so that the scan engine returns an ICAP 403 response instead of the 200 response that is demonstrated here.

```
S: ICAP/1.0 200 OK
S: ISTag: "1049478470"
S: Date: Fri Apr 04 17:59:29 2003 GMT
S: Service: Symantec AntiVirus Scan Engine/4.0.4.41
S: Service-ID: SYMCScan/4.0.4.41
S: X-Infection-Found: Type=0; Resolution=2;
Threat=Bloodhound.WordMacro;
S: X-Violations-Found: 1
    BHM97UR.DOC
    Bloodhound.WordMacro
    18950
    2
S: X-Outer-Container-Is-Mime: 0
S: Encapsulated: res-hdr=0, res-body=83
S:
S: HTTP/1.1 200 OK
S: Content-Length: 202
S: Pragma: no-cache
S: Content-Type: text/html
S:
S: ca
[ Chunk of size 202 bytes - access denied HTML message ]
S:
S: 0
S:
```

The response in the preceding section of code illustrates the following two X-headers that are used in responses from the scan engine when a problem is found during a scan:

- The X-Infection-Found header provides a brief description of a problem that is found during a scan. This header will be removed in a future release of the software.
- The X-Violations-Found header provides a complete manifest of any and all problems found during a scan. The first line is the total number of problems found (in this case, 1). For each problem that is identified, a block of detailed information follows (see the four lines of indented text shown in the example). The first line indicates the name of the file that contains the problem. If the infection is found in a file that is contained in a container file (such as a .zip file), this field contains the path to the specific file. The second line indicates the name of the problem or infection. The third line provides the problem ID or the virus ID. The fourth line indicates the resolution for the specific problem. In this case, the 2 indicates that the infection was not repaired.

Chapter 4

Constructing clients using the client API library

This chapter includes the following topics:

General procedure for scanning

The Symantec AntiVirus Scan Engine client API library provides a set of functions to simplify communication with the Symantec AntiVirus Scan Engine using ICAP.

The procedure for scanning data is as follows:

- The client calls ScanClientStartUp() once to initialize the client library and to set up the scheduling.
- Files are scanned using either ScanClientScanFile() or ScanClientStreamStart(), ScanClientStreamSendBytes() (as many times as necessary), and ScanClientStreamFinish().
- Information on infections found (if any) is obtained using ScanResultGetNumProblems() and ScanResultGetProblem().
- Completion of scanning for a particular file is indicated using ScanResultsFree().
- Completion of all scanning is indicated (for example, before program termination) using ScanClientShutDown() to free resources.

Sample code is included in the *Symantec AntiVirus Scan Engine Software Developer's Guide* for convenience. The sample code also is included in the distribution package. The sample program demonstrates how to use the Symantec AntiVirus Scan Engine API 192.168.1.2 to scan files.

See "Using the API" on page 73.

Compiling and linking

Table 4-1 describes requirements for compiling and linking on Solaris, Red Hat Linux, and Windows 2000 Server/Advanced Server.

Note: The code (in all versions of all supported platforms) is compiled with position-independent code so that it can be used in shared libraries.

Table 4-1 Compiling and linking requirements

Platform	Include	Initialize	Link
Solaris	#include "symcsapi.h"	none	libsocket.a libnsl.a (for example, gcc -lsocket -lnsl)

Table 4-1 Compiling and linking requirements

Platform	Include	Initialize	Link
Red Hat Linux	#include “symcsapi.h”	none	libnsl.a (for example, gcc -lnsl)
Windows 2000 Server/ Advanced Server	#include “symcsapi.h”	initialize winsock	winsock (for example, WS2_32.LIB)

Windows 2000 Server/Advanced Server

To compile on Windows 2000 Server/Advanced Server, use Microsoft Visual Studio 6 or 7.

The source code should #include the symcsapi.h file and link to SYMCSAPI.lib. The application should be compiled with multithreaded runtime libraries. For example, in Microsoft Visual Studio, under Project Settings, click the C/C++ tab, click Code Generation, and click Multithreaded Libraries.

Note: You must link to the winsock library (WS2_32.LIB) and initialize winsock in the source code before scanning files. You must release winsock when all network access is complete (usually just before exiting the program).

Initializing winsock

The client application must initialize winsock before scanning files. The following example demonstrates winsock initialization:

```
#include <windows.h>
.
.
.
    // start up winsock
    WORD wVersionRequested;
    WSADATA wsaData;
    int err;


    // Load WinSock, request version 1.0.
    wVersionRequested = MAKEWORD(1, 0);
    err = WSAStartup(wVersionRequested, &wsaData);
    if (err != 0)
    {
```

```
    // ERROR
}


// Confirm WinSock supports the version we requested.
if (LOBYTE(wsaData.wVersion) != 1 ||
        HIBYTE(wsaData.wVersion) != 0)
{
    WSACleanup();
    // ERROR
}
```

Shutting down winsock

You must release winsock when all network access is complete.

```
if (WSACleanup() == SOCKET_ERROR)
    ;// ERROR
```

Solaris

To compile on Solaris, use gcc compiler version 2.95 or 3.2.

The source code that makes calls to the library should #include the symcsapi.h header file. In the makefile (or command line) that compiles the program, libsymcsapi.a should be added and, if it is not already used, -lnsl -lsocket should be added. For example:

gcc mycode.c libsymcsapi.a -lsocket -lnsl

Note: If you are compiling your own multithreaded application, you must define the reentrant symbol (-D_REENTRANT) so that multithreading functions properly. The API libraries are already compiled in this manner.

Red Hat Linux

To compile on Red Hat Linux 7.2 and later, use gcc compiler version 2.95 or 3.2.

The source code that makes calls to the library should #include the symcsapi.h header file. In the makefile (or command line) that compiles the program, libsymcsapi.a should be added and, if it is not already used, -lnsl should be added. For example:

gcc mycode.c libsymcsapi.a -lnsl

Note: If you are compiling your own multithreaded application, you must define the reentrant symbol (-D_REENTRANT) so that multithreading functions properly. The API libraries are already compiled in this manner.

Exceptions and error handling

The Symantec AntiVirus Scan Engine API C library does not throw exceptions or include exception handling. Detected errors are returned as result codes from the functions.

API functions

The API functions are as follows:

- ScanClientStartUp
- ScanClientScanFile
- ScanResultGetNumProblems
- ScanResultGetProblem
- SC_DECODE_DISPOSITION
- ScanResultsFree
- ScanClientShutDown
- ScanClientStreamStart
- ScanClientStreamSendBytes
- ScanClientStreamFinish
- ScanClientStreamAbort

File-based scanning

If you are developing a file-based-scanning client, you will use the following functions:

- ScanClientStartUp()
- ScanClientScanFile()
- ScanResultGetNumProblems()
- ScanResultGetProblem()
- ScanResultsFree()
- ScanClientShutDown()

Stream-based scanning

If you are developing a stream-based-scanning client, you will use the following functions:

- ScanClientStartUp()
- ScanClientStreamStart()
- ScanClientStreamSendBytes()
- ScanClientStreamFinish()
- ScanClientStreamAbort()
- ScanResultGetNumProblems()
- ScanResultGetProblem()
- ScanResultsFree()
- ScanClientShutDown()

ScanClientStartUp

The ScanClientStartUp function lets a new client begin submitting files to the Symantec AntiVirus Scan Engine.

```
SC_ERROR ScanClientStartUp
(
    HSCANCLIENT    *client,
    LPSTR          pszClientConfiguration
)
```

ScanClientStartUp parameters

Table 4-2 lists the parameters that are used.

Table 4-2 ScanClientStartUp parameters

Parameter	Description
client	Address of an HSCANCLIENT variable. The HSCANCLIENT variable is a handle to a status data structure used throughout the scanning process. This variable contains information about the servers that are being used and the scheduling mechanism. Memory is allocated for this data structure by this call and must be freed using ScanClientShutDown() when scanning is completed.
pszClientConfiguration	A null-terminated string that contains configuration information. Entries are separated with three semicolons (;;;), and no spaces are allowed. Three options are currently supported: ■ Server:ipaddress:port;;;…: All Symantec AntiVirus Scan Engines that are used should be listed. At least one server must be listed. ■ FailRetryTime:<seconds>: If the client fails to connect to a Symantec AntiVirus Scan Engine, wait <seconds> before trying to connect to that server again (use only the other servers in the meantime, unless they have all failed recently). The default setting is 30 seconds. ■ ReadWriteTime:<seconds>: If after <seconds> no response is received from the scan engine (when data is being transmitted to the scan engine), or if the transmission of data does not complete (when data is being receiving from the scan engine), return an error message.

ScanClientStartUp return codes

Return codes are listed in Table 4-3. Negative values are warnings. Positive values are errors.

Table 4-3 ScanClientStartUp return codes

Value	Return code	Description
3	SC_MEMORY_ERROR	A memory allocation error has occurred.
1	SC_INVALID_PARAMETER	A parameter that was passed to the function was invalid.
0	SC_OK	Success.

ScanClientStartUp description

In a multithreaded environment, the client should make a single call to ScanClientStartUp on behalf of all of its threads. This way, all threads use scheduling via a single scheduler rather than multiple schedulers.

Parameter pszClientConfiguration should look like the following:

Server:1.2.3.4:1344;;;Server:1.2.3.5:1344

At least one Symantec AntiVirus Scan Engine must be specified. Entries are separated with three semicolons. No spaces are allowed. If the default port (1344) is used, the port number (and the colon) can be omitted.

Note: When multiple Symantec AntiVirus Scan Engines are specified, the API provides automatic scheduling across any number of scan engines. The API determines the appropriate scan engine to receive the next file to be scanned, based on the scheduling algorithm. If a scan engine is unreachable or stops responding during a scan, another scan engine is called and the faulty scan engine is taken out of rotation for a period of time (30 seconds is the default setting). If all of the scan engines are out of rotation, the faulty scan engines are called again. The API does not stop trying to contact the scan engines unless five engines do not respond or it appears that a file that is being scanned might have caused more than one engine to stop responding.

ScanClientScanFile

The ScanClientScanFile function is called to have the Symantec AntiVirus Scan Engine scan a file for viruses.

Note: The OriginalFileName parameter must be Unicode UTF-8 encoded.

```
SCSCANFILE_RESULT ScanClientScanFile
(
    HSCANCLIENT    hScanClient,
    LPSTR          pszOriginalFileName,
    LPSTR          pszActualFileName,
    LPSTR          pszOutputFileName,
    LPSTR          pszScanPolicy,
    LPHSCANRESULTS phScanResults
)
```

ScanClientScanFile parameters

Table 4-4 lists the parameters that are used.

Table 4-4 ScanClientScanFile parameters

Parameter	Description
hScanClient	An HSCANCLIENT variable that has been initialized. See "ScanClientStartUp" on page 54.
pszOriginalFileName	The name of the file to be scanned. (The original name of the file on the end user's computer.) This parameter is ignored if the PassFileByName=1 string is set in pszScanPolicy.
pszActualFileName	The name (path) of the file to scan on the Scan Client computer. This path may or may not be different than pszOriginalFileName. For example, an end user may upload a file to be scanned called sample.doc, but the same file may be stored on the ISP computer as temp123.doc. In this case, the pszOriginalFileName is sample.doc, but the pszActualFileName may be /tmp/temp123.doc.

Table 4-4 ScanClientScanFile parameters

Parameter	Description
pszOutputFileName	The storage location for the repaired file. (No output file is created unless the input file is infected and the infection is repairable.) This parameter can be one of the following: ■ A null-terminated string that is a path to where the repaired file is to be stored. ■ A char array at least MAX_STRING long, with the first byte set to '\0'. The API generates a file name for the repair file. When the function returns, pszOutputFileName is set to the name of the repaired file. **Note:** If this parameter is NULL, the API has the Symantec AntiVirus Scan Engine scan the file for viruses but not attempt repair. However, this is not the recommended method for forcing the scan engine to scan files for viruses but not attempt repair. Set the pszScanPolicy to ScanOnly instead. If local scanning options are set using pszScanPolicy, the Symantec AntiVirus Scan Engine ignores this parameter and repairs the file in place.

Table 4-4 ScanClientScanFile parameters

Parameter	Description
pszScanPolicy	A null-terminated string of the form <option:value>;;;<option:value>… No spaces are allowed. The following options are supported: ■ ScanOnly:1: Scan for viruses but do not attempt repair. ■ AlwaysReportDefInfo:1: If a clean file is scanned, a Problem Incident is created with only the virus definitions date and revision number. ■ RepairOnly:1: Attempt to repair infected files but do not delete files that cannot be repaired. ■ PassFileByName:1: The file to be scanned is on the same computer as the Symantec AntiVirus Scan Engine or can be accessed via NFS or another network file protocol. In this case, the file is not sent over a socket. Instead, the Symantec AntiVirus Scan Engine opens the file directly for scanning. If a repair is required, the repair is made in the local directory. If this string is set, the Symantec AntiVirus Scan Engine uses the pszActualFileName parameter to read the file and uses the ExtensionList= and ExclusionList= options to determine how to handle scanning for a specific file type. This functionality is not available for this release. See "Maximizing performance" on page 14.
phScanResults	When the function returns, this handle points at a structure that contains information about any problems (infections) found during the scan. If none were found, this parameter is NULL unless the AlwaysReportDefInfo:1 policy is used in the scan. Information regarding problems is extracted using the described functions. If this parameter is not NULL, the memory must be released using ScanResultsFree().

ScanClientScanFile return codes

Return codes are listed in Table 4-5. Negative values indicate that a virus was detected. Positive values are errors. Zero indicates a clean file.

Table 4-5 ScanClientScanFile return codes

Value	Return code	Description
-3	SCSCANFILE_INF_NO_REP	The file was infected with a virus, but no repair was possible.
-1	SCSCANFILE_INF_REPAIRED	The file was infected, and repair was successful.
0	SCSCANFILE_CLEAN	No virus was found.
1	SCSCANFILE_FAIL_CONNECT	Attempt to connect to a Symantec AntiVirus Scan Engine failed.
2	SCSCANFILE_FAIL_INPUTFILE	A problem was encountered reading the file to be scanned.
3	SCSCANFILE_FAIL_ABORTED	The scan was aborted abnormally.
4	SCSCANFILE_INVALID_PARAM	Function was called with an invalid parameter.
5	SCSCANFILE_FAIL_RECV_FILE	An error occurred when attempting to receive the repaired file.
6	SCSCANFILE_FAIL_MEMORY	A memory allocation error has occurred.
7	SCSCANFILE_FAIL_FILE_ACCESS	The server couldn't access the file to be scanned. This error usually occurs for LOCAL scans when the file permissions are wrong or when the file is not in the path that is specified in the LocalFileScanDir parameter on the server. This error can also occur when the API encounters a problem while writing repaired file data that was received from the scan engine to the output file.
10	SCSCANFILE_ERROR_SCANNING FILE	An internal server error occurred while the scan engine was attempting to repair the file.

Table 4-5 ScanClientScanFile return codes

Value	Return code	Description
15	SCSCANFILE_ABORT_NO_AV_SCANNING_LICENSE	No valid license for antivirus scanning functionality is installed.

ScanClientScanFile description

The ScanClientScanFile function determines the appropriate Symantec AntiVirus Scan Engine (when multiple Symantec AntiVirus Scan Engines are running) based on the scheduling algorithm. If a Symantec AntiVirus Scan Engine is unreachable or goes down during a scan, another server is called and the faulty server is taken out of rotation for a period of time. If all Symantec AntiVirus Scan Engines are out of rotation, the faulty servers are called again. The ScanClientScanFile function does not stop trying to contact a Symantec AntiVirus Scan Engine unless five servers are not functioning or it appears that a file being scanned might have caused two servers to go down.

ScanResultGetNumProblems

The ScanResultGetNumProblems function indicates the number of infections that are contained in an HSCANRESULT after scanning a file. Use the ScanResultGetProblem() function to get information about the infections that are reported in an HSCANRESULT after scanning a file.

```
SC_ERROR ScanResultGetNumProblems
(
    HSCANRESULT hScanResult,
    int *nNumProblems
)
```

ScanResultGetNumProblems parameters

Table 4-6 lists the parameters that are used.

Table 4-6 ScanResultGetNumProblems parameters

Parameter	Description
hScanResult	An HSCANRESULT variable that is returned by ScanClientScanFile() or ScanClientStreamFinish().
nNumProblems	When the function returns, this parameter is set to the number of infections in the scanned file. If the AlwaysReportDefInfo:1 scan policy option is used, at least one (blank) incident is reported, along with the virus definitions date and revision number.

ScanResultGetNumProblems return codes

Return codes are listed in Table 4-7. Negative values are warnings. Positive values are errors.

Table 4-7 ScanResultGetNumProblems return codes

Value	Return code	Description
0	SC_OK	Success.
-1	SC_NULL_PARAMETER	A parameter that was passed to the function is NULL when it shouldn't be.

ScanResultGetProblem

The ScanResultGetProblem function is used to get specific information about a virus.

```
SC_ERROR ScanResultGetProblem
(
    HSCANRESULT hScanResult,
    int nProblemNum,
    int nAttribute,
    LPSTR pszValueOut,
    LPINT pnValueLengthInOut
)
```

ScanResultGetProblem parameters

Table 4-8 lists the parameters that are used.

Table 4-8 ScanResultGetProblem parameters

Parameter	Description
hScanResult	An HSCANRESULT variable that is returned by ScanClientScanFile() or ScanClientStreamFinish().
nProblemNum	Integer specifying which problem entry is being investigated.

Table 4-8 ScanResultGetProblem parameters

Parameter	Description
nAttribute	Identifies which attribute is being queried. Valid attributes are: ■ SC_PROBLEM_FILENAME: The file name in which the infection occurred. ■ SC_PROBLEM_VIRUSNAME: The name of the virus that has infected the file. ■ SC_PROBLEM_VIRUSID: The unique numerical ID of the virus. ■ SC_PROBLEM_DISPOSITION: Problem resolution. This value is a number (in a string format) that indicates whether the virus was cleaned from the file. ■ SC_PROBLEM_DEFINITION_DATE: The date stamp on the virus definitions. ■ SC_PROBLEM_DEFINITION_REV: Revision number of the definitions.
pszValueOut	Buffer that holds the attribute value being retrieved.
pnValueLengthInOut	Pointer to variable that holds the size of the pszValueOut buffer. When the function returns, the size of the attribute string is placed in this variable. You can determine if the buffer was large enough to hold the entire string by seeing whether pnValueLengthInOut is smaller after the call than before (if the value is smaller, the buffer was large enough).

ScanResultGetProblem return codes

Return codes are listed in Table 4-9. Negative values are warnings. Positive values are errors.

Table 4-9 ScanResultGetProblem return codes

Value	Return code	Description
5	SC_OUTOFRANGE_PARAMETER	At least one parameter that was passed to the function was out of range.
1	SC_INVALID_PARAMETER	A parameter that was passed to the function was invalid.
0	SC_OK	Success.

ScanResultGetProblem description

A buffer is supplied to hold the information about the virus, and a pointer is supplied to an integer that holds the size of the buffer. When the function returns, the integer holds the amount of data that was placed in the buffer. If the buffer is not large enough to hold the information, as much information as possible is copied into the buffer. If the value of the integer is the same after the call as it was before the call, the buffer most likely was not large enough to hold the information. See the description of the nAttribute parameter for the type of information that can be retrieved.

SC_DECODE_DISPOSITION

The problem disposition is retrieved with ScanResultGetProblem() in the form of a string. The SC_DECODE_DISPOSITION function is a macro that converts the string to an integer and defines the result codes as integers for easier processing.

```
int SC_DECODE_DISPOSITION( char *pszDisposition )
```

SC_DECODE_DISPOSITION parameters

Table 4-10 lists the parameters that are used.

Table 4-10 SC_DECODE_DISPOSITION parameters

Parameter	Description
pszDisposition	The pszValueOut that is returned from ScanResultGetProblem with nAttribute equal to SC_PROBLEM_DISPOSITION.

SC_DECODE_DISPOSITION return codes

Return codes are listed in Table 4-11.

Table 4-11 SC_DECODE_DISPOSITION return codes

Return code	Description
SC_DISP_REPAIRED	The infected file was repaired.
SC_DISP_UNREPAIRED	The infected file was not repaired.
SC_DISP_DELETED	The infected file was deleted.

ScanResultsFree

The ScanResultsFree function is used to free the HSCANRESULT structure. This function must be called when the result structure is no longer needed to free the allocated memory.

```
SC_ERROR ScanResultsFree( HSCANRESULT hScanResult )
```

ScanResultsFree return codes

Return codes are listed in Table 4-12. Negative values are warnings. Positive values are errors.

Table 4-12 ScanResultsFree return codes

Value	Return code	Description
0	SC_OK	Success.
-1	SC_NULL_PARAMETER	A parameter that was passed to the function is NULL when it shouldn't be.

ScanClientShutDown

The ScanClientShutDown function is called to clean up after all scanning is complete (for example, before program termination).

```
SC_ERROR ScanClientShutDown( HSCANCLIENT hScanClient )
```

ScanClientShutDown return codes

Return codes are listed in Table 4-13. Negative values are warnings. Positive values are errors.

Table 4-13 ScanClientShutDown return codes

Value	Return code	Description
3	SC_MEMORY_ERROR	A memory allocation error has occurred.
0	SC_OK	Success.

ScanClientStreamStart

The ScanClientStreamStart function is used for scanning streams. It can also be used when you want to accept an input stream for scanning rather than an entire file at one time. For example, if a file is being received via an HTTP stream as a user uploads a file to a Web site, stream scanning can be used.

Note: The OriginalFileName parameter must be Unicode UTF-8 encoded.

```
SC_ERROR ScanClientStreamStart
(
    HSCANCLIENT hScanClient,
    LPSTR       pszOriginalFileName,
    LPSTR       pszScanPolicy,
    HSCANSTREAM *phScanStream
)
```

ScanClientStreamStart parameters

Table 4-14 lists the parameters that are used.

Table 4-14 ScanClientStreamStart parameters

Parameter	Description
hScanClient	This parameter should be set up using ScanClientStartUp().
pszOriginalFileName	The original name of the file to be scanned as it was named on the end user's computer.
pszScanPolicy	A null-terminated string of the form <option:value>;;;<option:value>… No spaces are allowed. See "ScanClientScanFile" on page 57.
phScanStream	Pointer to an HSCANSTREAM variable.

ScanClientStreamStart return codes

Return codes are listed in Table 4-15. Negative values are warnings. Positive values are errors.

Table 4-15 ScanClientStreamStart return codes

Value	Return code	Description
6	SC_CONNECT_FAILURE	Attempt to connect to a Symantec AntiVirus Scan Engine failed.
3	SC_MEMORY_ERROR	A memory allocation error has occurred.
2	SC_SOCKET_FAILURE	A socket communications error has occurred.
1	SC_INVALID_PARAMETER	A parameter that was passed to the function was invalid.
0	SC_OK	Success.

Setting up stream scanning

The stream scanning feature lets the stream from the end user's computer be sent directly to the Symantec AntiVirus Scan Engine, rather than first receiving the entire file and then calling ScanClientScanFile().

To set up stream scanning

1. Call ScanClientStreamStart() to initialize the HSCANSTREAM variable.
   ScanClientStreamStart() must be called for each file to be scanned to initialize the HSCANSTREAM variable. HSCANSTREAM variables can be reused only after the stream has been closed with ScanClientStreamFinish() or ScanClientStreamAbort(). The OriginalFileName parameter must be Unicode UTF-8 encoded.
2. Send data to the server in chunks as it is received using ScanClientStreamSendBytes().
3. When all data is sent, call ScanClientStreamFinish().
   To abort the scan between the …Start() and …Finish() calls, call ScanClientStreamAbort().

ScanClientStreamSendBytes

The ScanClientStreamSendBytes function is used to send chunks of data after HSCANSTREAM has been initialized with ScanClientStreamStart().

See "ScanClientStreamStart" on page 66.

```
SC_ERROR ScanClientStreamSendBytes
(
    HSCANSTREAM hStream,
    LPBYTE      lpabyData,
    DWORD       dwLength
)
```

ScanClientStreamSendBytes parameters

Table 4-16 lists the parameters that are used.

Table 4-16 ScanClientStreamSendBytes parameters

Parameter	Description
hStream	The HSCANSTREAM variable, which must be initialized by a call to ScanClientStreamStart().
lpabyData	Pointer to a buffer that contains the next chunk of data to be sent.
dwLength	Size, in bytes, of the next chunk of data to be sent.

ScanClientStreamSendBytes return codes

Return codes are listed in Table 4-17. Negative values are warnings. Positive values are errors.

Table 4-17 ScanClientStreamSendBytes return codes

Value	Return code	Description
2	SC_SOCKET_FAILURE	A socket communications error has occurred.
1	SC_INVALID_PARAMETER	A parameter that was passed to the function was invalid.
0	SC_OK	Success.

ScanClientStreamFinish

The ScanClientStreamFinish function must be called after an entire file has been sent to the Symantec AntiVirus Scan Engine to be scanned using ScanClientStreamSendBytes().

See "ScanClientStreamStart" on page 66.

```
SCSCANFILE_RESULT ScanClientStreamFinish
(
    HSCANSTREAM    hStream,
    LPSTR          pszOutputFileName,
    LPHSCANRESULTS phScanResults
)
```

ScanClientStreamFinish parameters

Table 4-18 lists the parameters that are used.

Table 4-18 ScanClientStreamFinish parameters

Parameter	Description
hStream	The HSCANSTREAM variable, which must be initialized by a call to ScanClientStreamStart().
pszOutputFileName	The storage location for the repaired file. (No output file is created unless the input file is infected and the infection is repairable.) This parameter can be one of the following: ■ A null-terminated string that is a path to where the repaired file is to be stored. ■ A char array at least MAX_STRING long, with the first byte set to '\0'. The API generates a file name for the repair file. When the function returns, pszOutputFileName is set to the name of the repaired file. **Note:** If this parameter is NULL, the API has the Symantec AntiVirus Scan Engine scan the file for viruses but not attempt repair. However, this is not the recommended method for forcing the scan engine to scan files for viruses but not attempt repair. Set the pszScanPolicy to ScanOnly instead.

Table 4-18 ScanClientStreamFinish parameters

Parameter	Description
phScanResults	When the function returns, this handle points at a structure that contains information about any problems (infections) found during the scan. If none were found, this parameter is NULL unless the AlwaysReportDefInfo:1 policy is used in the scan. Information regarding problems is extracted using the described functions. If this parameter is not NULL, the memory must be released using ScanResultsFree().

ScanClientStreamFinish return codes

Return codes are listed in Table 4-19. Negative values are warnings. Positive values are errors.

Table 4-19 ScanClientStreamFinish return codes

Value	Return code	Description
-3	SCSCANFILE_INF_NO_REP	The file was infected with a virus, but no repair was possible.
-1	SCSCANFILE_INF_REPAIRED	The file was infected, and repair was successful.
0	SCSCANFILE_CLEAN	No virus was found.
3	SCSCANFILE_FAIL_ABORTED	The scan was aborted abnormally.
4	SCSCANFILE_INVALID_PARAM	Function was called with an invalid parameter.
5	SCSCANFILE_FAIL_RECV_FILE	An error occurred when attempting to receive the repaired file.
6	SCSCANFILE_FAIL_MEMORY	A memory allocation error has occurred.

Table 4-19 ScanClientStreamFinish return codes

Value	Return code	Description
7	SCSCANFILE_FAIL_FILE_ACCESS	The server couldn't access the file to be scanned. This error usually occurs for LOCAL scans when the file permissions are wrong or when the file is not in the path that is specified in the LocalFileScanDir parameter on the server. This error can also occur when the API encounters a problem while writing repaired file data that was received from the scan engine to the output file.
10	SCSCANFILE_ERROR_SCANNING FILE	An internal server error occurred while the scan engine was attempting to repair the file.
15	SCSCANFILE_ABORT_NO_AV_ SCANNING_LICENSE	No valid license for antivirus scanning functionality is installed.

ScanClientStreamAbort

The ScanClientStreamAbort function is called to abort a scan between calls to ScanClientStreamStart() and ScanClientStreamFinish().

```
SC_ERROR ScanClientStreamAbort
(
    HSCANSTREAM hStream
)
```

ScanClientStreamAbort return codes

Return codes are listed in Table 4-20. Negative values are warnings. Positive values are errors.

Table 4-20 ScanClientStreamAbort return codes

Value	Return code	Description
1	SC_INVALID_PARAMETER	A parameter that was passed to the function was invalid.
0	SC_OK	Success.
-1	SC_NULL_PARAMETER	A parameter that was passed to the function is NULL when it shouldn't be.

Appendix A

Using the API

This chapter includes the following topics:

About the sample code

The sample code shown in the *Symantec AntiVirus Scan Engine Software Developer's Guide* is for convenience. Sample code also is included on the distribution CD. This sample program demonstrates how use the Symantec AntiVirus Scan Engine API to scan files.

This example demonstrates the use of both file-based and stream-based scanning. File-based scanning is enabled by default.

Sample code

The usage and syntax is as follows:

example <Scan Engine ip>:<Scan Engine port> <input file> [<input file>...]

On Solaris, compile the code using code similar to the following:

```
cc example.cpp libsymcsapi.a -lsocket -lnsl -DUNIX
```

On Windows, link to the winsock library. For example:

```
ws2_32.lib
```

```
#if defined( WIN32 )
#pragma warning(push,3)
#endif
```

```
#include <stdio.h>
#include <string>
#include "symcsapi.h"
```

```
#if defined( WIN32 )
#include <windows.h>
#endif
```

```
#if defined( WIN32 )
#include <windows.h>
#endif
```

```
// Function Prototypes
void print_prob_info( HSCANRESULTS hResults, int iWhichProblem );
int scanfile( HSCANCLIENT scanclient, char *orig_name, char
    *actual_name);
```

```
int main( int argc, char *argv[])
{
    HSCANCLIENT scanclient=NULL;
    char pszStartUpString[MAX_STRING];
    int i;
```

```
#if defined( WIN32 )
    // start up winsock
    WORD wVersionRequested;
```

```
    WSADATA wsaData;
    int err;

    // Load WinSock, request version 1.0.
    wVersionRequested = MAKEWORD(1, 0);
    err = WSAStartup(wVersionRequested, &wsaData);
    if (err != 0)
    {
            return 3;// ERROR
    }

    // Confirm WinSock supports the version we requested.
    if (LOBYTE(wsaData.wVersion) != 1 ||
        HIBYTE(wsaData.wVersion) != 0)
    {
        WSACleanup();
        return 3;// ERROR
    }

#endif // defined( WIN32 )

    if( argc < 3 )
    {
        printf( "Usage: %s <ipaddress>:<port> <input-file> [<input-
            file>...]\n", argv[0] );
        return 1;
    }

    sprintf( pszStartUpString, "server:%s", argv[1] );

    if( ScanClientStartUp( &scanclient, pszStartUpString ) > 0 )
    {
        printf( "Error in ScanClientStartUp\n");
        return 1;
    }

    for( i=2; i<argc; i++ )
    {
        printf( "============================\n");
        printf( "Scanning file: %s\n", argv[i] );
        printf( "============================\n");
        scanfile( scanclient, argv[i], argv[i] );
    }

    ScanClientShutDown( scanclient );

#if defined( WIN32 )
    if (WSACleanup() == SOCKET_ERROR)
    {
        // ERROR
    }
#endif

        return 0;
}
```

```
/*
** Scans a file
**
** Parameters:
**   scanclient: Scan Engine client connection
**   orig_name: Original name of the file
**   actual_name: Name of the file on this machine
**
** Returns:
**   1 for success
**   0 for failure
*/
int scanfile( HSCANCLIENT scanclient, char *orig_name, char
    *actual_name)
{
    HSCANRESULTS results=NULL;
    char repair_file[MAX_STRING];
    int numproblems=0;

    // Set up the buffer to accept the repair filename
    repair_file[0] = 0;

#define SCANWHOLEFILE
#if defined(SCANWHOLEFILE)
    // Perform the scan
    SCSCANFILE_RESULT answer = ScanClientScanFile( scanclient,
        orig_name,
        actual_name,
        repair_file,
        "",
        &results
        );
#else // SCANWHOLEFILE
    char sendbuff[8 * 1024];

    HSCANSTREAM hScanStream = NULL;
    if (SC_OK != ScanClientStreamStart(scanclient, orig_name, "",
            &hScanStream))
        return 0;

    // Open the file and send it
#if defined (WIN32)
    HANDLE fd = CreateFile( actual_name, GENERIC_READ, 0, NULL,
            OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
    if (fd == INVALID_HANDLE_VALUE)
    {
        ScanClientStreamAbort(hScanStream);
        return 0;
    }

    DWORD iChunkSize;
    if (!ReadFile( fd, (void *) sendbuff, sizeof(sendbuff),
            &iChunkSize, NULL))
    {
```

```
            CloseHandle( fd );
            ScanClientStreamAbort(hScanStream);
            return 0;
        }

        if (iChunkSize < 0)
        {
            CloseHandle( fd );
            ScanClientStreamAbort(hScanStream);
            return 0;
        }

        while ( iChunkSize > 0 )
        {
            if (SC_OK != ScanClientStreamSendBytes(hScanStream,
                    (LPBYTE)sendbuff, iChunkSize))
            {
                CloseHandle( fd );
                // Do not call ScanClientStreamAbort here
                return 0;
            }

            // Keep reading the file
            if (!ReadFile( fd, (void *) sendbuff, sizeof(sendbuff),
                    &iChunkSize, NULL))
            {
                CloseHandle( fd );
                ScanClientStreamAbort(hScanStream);
                return 0;
            }

            if (iChunkSize < 0)
            {
                CloseHandle( fd );
                ScanClientStreamAbort(hScanStream):
                return 0;
            }
        }
        CloseHandle( fd );
    #else // if defined(WIN32)
        int fd = open(actual_name, O_RDONLY);
        if(fd < 0)
        {
            ScanClientStreamAbort(hScanStream);
            return 0;
        }

        int iChunkSize = read( fd,  sendbuff, sizeof( sendbuff));

        if( iChunkSize < 0 )
        {
            close( fd );
            ScanClientStreamAbort(hScanStream);
            return 0;
        }
```

```
    while ( iChunkSize > 0 )
    {
        if (SC_OK != ScanClientStreamSendBytes(hScanStream,
                (LPBYTE)sendbuff, iChunkSize))
        {
            close( fd );
            // Do not call ScanClientStreamAbort here
            return 0;
        }

        // Keep reading the file
        iChunkSize = read( fd,  sendbuff, sizeof( sendbuff));
        if( iChunkSize < 0 )
        {
            close( fd );
            ScanClientStreamAbort(hScanStream);
            return 0;
        }
    }

    close( fd );
#endif // if defined(WIN32)

    SCSCANFILE_RESULT answer = ScanClientStreamFinish(hScanStream,
            repair_file, &results);
#endif // SCANWHOLEFILE

    if( answer > 0 )
    {
        printf( "**** ERROR! Couldn't scan file\n");
        return 0;
    }

    switch( answer )
    {
    case SCSCANFILE_INF_NO_REP:
        printf( "File is infected and cannot be repaired\n");
        break;
    case SCSCANFILE_INF_REPAIRED:
        printf( "File was infected, and has been repaired\n");
        break;
    case SCSCANFILE_CLEAN:
        printf( "File is clean\n");
        break;
    default:
        printf( "ScanClientScanFile returned an unexpected
            value\n");
        break;
    }

    // The results structure will be non-null if a virus was
            detected.
    if( results )
    {
        if( strlen( repair_file ) )
```

```
        {
            printf( "Repaired file saved as: %s\n", repair_file );
        }
        else
        {
            printf( "No repair file generated\n");
        }

        if( ScanResultGetNumProblems( results, &numproblems ) > 0 )
        {
            printf( "Error getting number of problems\n");
            return 2;
        }
        printf( "%s had %d infection(s):\n",actual_name,
                numproblems );
    }
    else
    {
        numproblems = 0;
    }

    for( int i=0; i<numproblems; i++ )
    {
        print_prob_info( results, i );
    }

    // Be sure to free the results when done!
    ScanResultsFree( results );

    return 1;
}

/*
** Prints scan related information
**
** Parameters:
**   hResults: structure containing the scan results
**   iWhichProblem: scan file problem ID
*/
void print_prob_info( HSCANRESULTS hResults, int iWhichProblem )
{
    char attrib[MAX_STRING];
    int attrib_size;
    int iDisposition;

    attrib_size = MAX_STRING;
    ScanResultGetProblem( hResults,
        iWhichProblem,
        SC_PROBLEM_FILENAME,
        attrib,
        &attrib_size );
    printf( "File Name: %s\n", attrib );

    attrib_size = MAX_STRING;
    ScanResultGetProblem( hResults,
```

```
        iWhichProblem,
        SC_PROBLEM_VIRUSNAME,
        attrib,
        &attrib_size );
    printf( "Virus Name: %s\n", attrib );

    attrib_size = MAX_STRING;
    ScanResultGetProblem( hResults,
        iWhichProblem,
        SC_PROBLEM_VIRUSID,
        attrib,
        &attrib_size );
    printf( "Virus ID: %s\n", attrib );

    attrib_size = MAX_STRING;
    ScanResultGetProblem( hResults,
        iWhichProblem,
        SC_PROBLEM_DISPOSITION,
        attrib,
        &attrib_size );

    iDisposition = SC_DECODE_DISPOSITION( attrib );
    switch( iDisposition )
    {
    case SC_DISP_UNREPAIRED:
        printf( "This infection could not be repaired\n");
        break;
    case SC_DISP_REPAIRED:
        printf( "This infection was repaired\n");
        break;
    case SC_DISP_DELETED:
        printf( "The file with this infection should be deleted\n");
        break;
    default:
        printf( "Unknown Disposition\n");
        break;
    }

    attrib_size = MAX_STRING;
    ScanResultGetProblem( hResults,
        iWhichProblem,
        SC_PROBLEM_DEFINITION_DATE,
        attrib,
        &attrib_size );
    printf( "Virus Definitions dated: %s\n", attrib );

    attrib_size = MAX_STRING;
    ScanResultGetProblem( hResults,
        iWhichProblem,
        SC_PROBLEM_DEFINITION_REV,
        attrib,
        &attrib_size );
    printf( "Virus Definitions Revision: %s\n", attrib );
}
```

Index

I

L

M

O

P

R

S

U

V

W